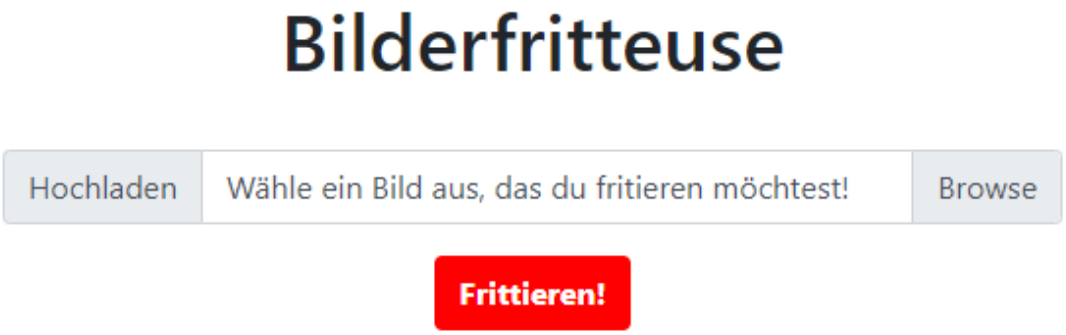
Bilderfritteuse Dokumentation

Nach Scheitern der Instandsetzung des Deppyers, musste ich nun auf eine API zurückgreifen, die weniger Funktionen bietet. Zuerst aber, generierte ich mit ChatGPT eine simple Startseite, in der ein Upload Feld und ein Frittieren Button angezeigt wurde.



Danach informierte ich mich darüber, was Composer ist, da meine neue Library Composer benötigte. Es stellt sich heraus, dass Composer Libraries direkt in das Repository importieren kann und alles konfiguriert. Nach ein paar Versuchen, die Library am richtigen Ort zu importieren, habe ich es dann auch geschafft. Ich generierte nun das Formular für die Fry.php klasse mit ChatGPT und probierte das neue Bild auch gleich anzuzeigen. Nun stiess ich aber auf einen Error. Das Bild wurde nicht angezeigt und der alt-text wurde verwendet. Das heilige Orakel ChatGPT wurde nochmals konsultiert, aber es konnte mir nicht weiterhelfen.

Nach einer Stunde gab ich um 16.15 frustriert auf du schob das Projekt bis zum Samstag wieder auf. Nun sah ich in den Kriterien, dass ich ein Projektplanungstool verwenden muss, und erstellte dementsprechend ein Kanban Board. Nach weiteren 1.5 Stunden fand ich dann endlich heraus, dass Xampp aus mir bisher immer noch nicht klaren Gründen den von mir vorhergesehenen «src» Ordner nicht accessen kann, deshalb verschob ich alle Dateien sowie die Bilder in meinen Htdocs Folder. Damit wurde ein Workaround geschaffen und ich konnte weiterarbeiten. Nun addete ich eine Scrollbar mit der sich die Qualität dynamisch ändern lässt. Um das dynamische Verändern zu gewährleisten, kopierte ich Javascript code von Stack Overflow und passte ihn an.

```
echo "<script>
    var qualitySlider = document.getElementById('qualitySlider');
    var qualityValue = document.getElementById('qualityValue');
    qualitySlider.oninput = function() {
        qualityValue.innerHTML = this.value;
    };
    qualitySlider.onchange = function() {
        var quality = this.value;
        var image = new Image();
        image.src = '$improved_file_path';
        image.onload = function() {
            var canvas = document.createElement('canvas');
            var ctx = canvas.getContext('2d');
            canvas.width = image.width;
            canvas.height = image.height;
            ctx.drawImage(image, 0, 0);
            var dataUrl = canvas.toDataURL('image/jpeg', quality / 100);
            var imgElement = document.querySelector('.responsive-image');
            imgElement.src = dataUrl;
        };
    };
</script>";
```

Danach fügte ich eine Dateivalidierung (Muss JPG und unter 200KB sein) und 3 Download Buttons hinzu. Das JPG konnten wir einfach so herunterladen, während das GIF und das PNG erstmal durch die GD Library konvertiert werden musste. Dazu nutzte ich die imagegif und imagepng Methoden.

```
if ($outputFormat == 'gif') {
    imagegif($im, $outputPath);
} elseif ($outputFormat == 'png') {
    imagepng($im, $outputPath);
}
```

Nun musste ich nur noch die Galerie bereitstellen und danach war das Projekt auch schon fertig. In der „galery.php“ (Typo) probierte ich erst die Bilder in einem Grid wie auf Instagram darzustellen und verwendete dafür ChatGPT, aber es funktionierte nicht. Nun addete ich noch mithilfe von ChatGPT ein Script, dass den Dateinamen bei einem Upload in der Uploadbar anzeigt, da die Darstellung zuvor ein wenig verwirrend war. Als ich dann auf Bootstrap das Imagecarousell gefunden habe, entschloss ich mich dazu, dies zu verwenden. Ich stellte noch sicher, dass die Bilder dem Viewport angepasst werden, und danach validierte ich den HTML code.

Den Code kann man unter folgendem Repository finden:

https://github.com/Fridadacat/Deepfryer/tree/main

Und folgend noch das Kanban-Board:

Ensi.kanbantool.com/b/999179-deepfryer

Login: User; Ensi, Passwort; bbzblprojekt